Multi- Agent Modeling of Societal Development and Cultural Evolution

Yidan Chen

June 12, 2006

Abstract

In the study of social sciences within human history, the model should reflect the historical process of development. Many agent based modeling applications currently exist, but approach the simulation from the aspect of the result. This project will instead implement the "bottom-up" approach by implementing the background world and individualized agents with specific attributes, using the Multi-Agent Simulator of Neighborhoods (MASON) library, to parallelize default societal and human characteristics. The importation of simple rules to regulate agent behavior establish the agent-based simulations with correlating results to historically recorded migration developments. The simulation records the complex behavior results through economic aspects of wealth distribution and trade, and social aspects of reproduction, death, health, and cultural exchange. The Sugarscape model has been previously implemented in Ascape by the Brookings Institute. In the spirit of innovation in the area of agent-based simulation, this project uses the MASON library from GMU and features MASON's 3D graphical option. The study results include graphical representations for the change in wealth distribution and patterns of agent migration in the iteration of the program.

1 Introduction

1.1 Background

The history of agent-based modeling begins with the invention of the von Neumann machines, which are not themselves agent-based models. In the late 1940s, John von Neumann suggested a machine which reproduced by following certain instructions on how to copy itself. The machine would then give its offspring a copy of its instructions, creating a "fertile offspring". Von Neumann drew up the machine as a grid containing a collection of cells,

creating the first of what later became known as cellular automata (5). The implication of the von Neumann machines is that the basis of life was information.

The concept of life as information spurred on research in the area of artificial life, most notably John Conway's game of Life. The game of Life operated under much simpler rules than those of the von Neumann machines. Life occurs on a virtual checkerboard, each square representing a cell in one of two states, either alive or dead. A new cell may be born with the condition of a certain number of neighbors and alive cells may die under the condition of certain numbers of neighbors. Conway initially reiterated the game by hand to study the variety of configurations. Later teams used computers to program the game in what became the first known case of artificial life from computers. Among the many questions that Life raised was whether a finite initial configuration can generate an infinite population (4). The implications of the simple rules were complex, leading programmers to wonder whether the such simple rules can apply to the real world to explain complex results.

From the implications of Life, bottom-up models of real world situations (1) followed in the form of Craig Reynolds's "boids" and later Robert Axtell and Joshua M. Epstein's Sugarscape (3). Boids stood for "birdoids", which investigated Reynold's theory for bird flocking through a computational model of decentralized activity. Rather than following a form of centralized control, each individual boid followed simple logical rules for its own good. The results showed a behavior that copied the movement of flocking as it occurred in nature. The results of the boids study further implicated that seemingly centralized activity in the real world are simply patterns created through the summation of the decentralized activity of individual agents or cells.

The boids study then brings agent-based modeling to the world of Sugarscape, initially created by Brookings researchers Joshua Epstein and Robert Axtell to study the aggregate effects of resulting interactions from simple rules on Sugarscape's agents. Sugarscape has been used to model a variety of complex situations, such as proto-history, cultural evolution, population pressures, and warfare, in the bottom-up approach with simple causes for

individual Agents.

This project is based on the original Sugarscape, outlined in Growing Artificial Societies (2). The code of the original Sugarscape is written using Ascape. Other developers, such as Tony Bigbee, have implemented the project in other configurations (ie: MASON, NetLogo) as well. Other forms of agent-based modeling systems (ie: RePast, Swarm) also exist, however the system currently in development is MASON from GMU.

1.2 The World

The basic logic behind the Sugarscape world is: "First there was sugar" (2), perhaps a biblical reference to "Let there be light". The world consists of a supposed bagel shaped planet with natives known as "agents". The agents go about their lives gathering sugar, reproducing and migrating for survival. Eventually the natives develop tribes and culture, accumulate wealth distribution, implement trade actions, and wage war for self and societal benefits. The ground is covered with sugar, ranging from a value of 0 for the least fertile land to a value of 4 for the greatest amount of sugar. The areas of different amounts of sugar are illustrated using different degrees of yellow.

For the 2D version of Sugarscape (Figure 1), the land is entirely in a window of 400 X 400 pixels, with two poles of maximum sugar on the upper left and lower right corners. Each pixel of the window is equivalent to a cell in one of two data grids: valgrid and agentgrid. The sugar value for the specific cell is stored in valgrid, which is a IntGrid2D. The IntGrid2D class is a MASON class that is similar to a 2D integer matrix. The valgrid stores the individual value for sugar as a variable in its integer field. The agentgrid is a SparceGrid2D for the 2D Sugarscape, which stores objects in its field rather than integers. The field allows for multiple agents in the same position.

The 3D Sugarscape (Figure 2) features different altitudes to distinguish between the different levels of sugar. The window has size 100 X 100 X 100

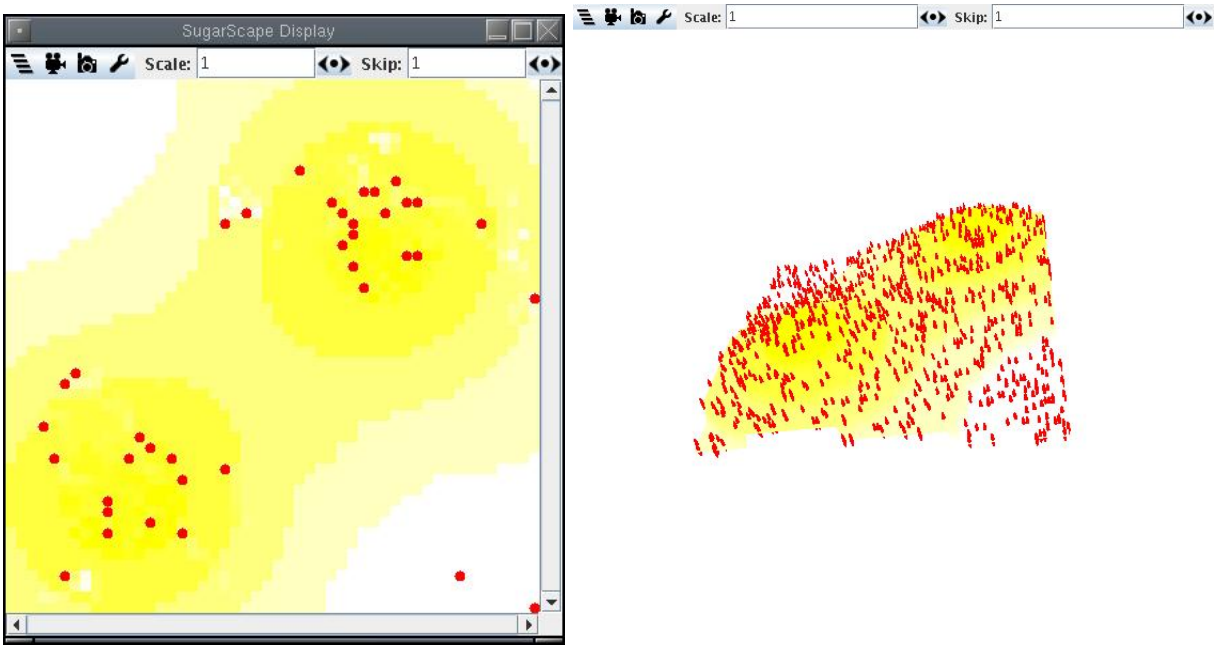

Figure 1: a:2D version, b: 3D version.

1.3 The Rules

The implementation of simple rules in the Sugarscape regulate behavior for agents and for the environment. Simple rules include the subcategories of agent-environment rules which mark the interaction of the agent with the environment, environment-environment rules which regulates environmental changes with cellular automata, and agent-agent rules that govern interactions between agents. The first and last category applies to Sugarscape in this the parameters of this project.

1.3.1 Finite Growth Rule

The sugar of the environment grows back at a finite rate of growthRate, set to 1 sugar unit per step. Unless the sugar level of a cell is the maximum level that the cell can hold, according to the map of sugar levels in "sugarfield.txt", the program will set the level to increment by the growthRate. In the case that the sugar level is the maximum of the cell, the program will set the level to equal the cell by default. The program extracts the data

fields from Sugarscape by setting up a Sugarscape state equal to the original Sugarscape, and using the state to call the grids.

1.3.2 Infinit Growth Rule

The infinite rule differs from the finite rule in that the sugar level of each cell returns to its maximum level with each iteration of the program. With each step, the program tests if a cell is below its optimal sugar level, and sets the cell to the maximum level allowed by the sugar map. The rule reiterates the process for each cell per each step during runtime.

1.3.3 Distribution Rule

The distribution rule was implemented for the purpose of tracking the economic distribution of the agents. The rule extracts the sugar level of each agent through the agent grid, and makes mathematical analysis to determine the distribution. The different levels of wealth are set to 0-10, 10-20, 20-30, 30-40, and 40 and more. The rule should demonstrate the economic phenomenon of a large population at the bottom of level versus a small population at the highest level. The rule should also show that a few percentage of the population controls the majority of the resources in a society.

1.4 The Agents

The agents of Sugarscape are individual entities, each with randomly assigned attributes in the aspects of metabolism, vision and initial sugar level. The agents' goal is to find more sugar and to collect the sugar. In the event that the Agent consumes more sugar than it collects, it will be eliminated from the board.

1.4.1 Sugar

The agents stay alive by the consummation of sugar in Sugarscape. The sugar is symbolic of natural resources on which humans rely. Similar to humans, agents act individually in their

collection of sugar, each acting to its own best interest. The agent searches for the best cell, defined as the cell with the most sugar and within the smallest distance from the agent's current position. The agent collects all of the sugar available at a cell with each movement, and adds the new amount of sugar to the amount in its storage, under double mySugar. With each step, the agent also consumes an amount of sugar equal to its metabolism, which is the value subtracted from mySugar. If an agent is not able to collect sugar matching or exceeding its metabolism, the agent is no longer accounted for on the grid, representing death of the agent in the system.

1.4.2 Migration

Two forms of migration were implemented: the Van Neumann neighborhood and the Moore neighborhood. Initially the program used the Moore neighborhood, and the agent checked cells in the eight surrounding directions before changing position. The Moore neighborhood was then replaced by the Van Neumann neighborhood, in which the agent only checks four directions and not the diagonal directions when it considers a change in position. The change was due to the need to decrease the difficulty in engineering rather than a functionality problem.

2 Testing

2.1 Functionality

The functionality testing consists of the change in parameters during runtime, through the variables in the Applet window of MASON. While the initial values of the variables are set by the program, the user can change the values for MaxVision, gridHeight, gridWidth and scale, agentCount and randomMovementProbability. The testing is done by changing the various parameters during runtime and refreshing the program to start() again (Figure 2a).

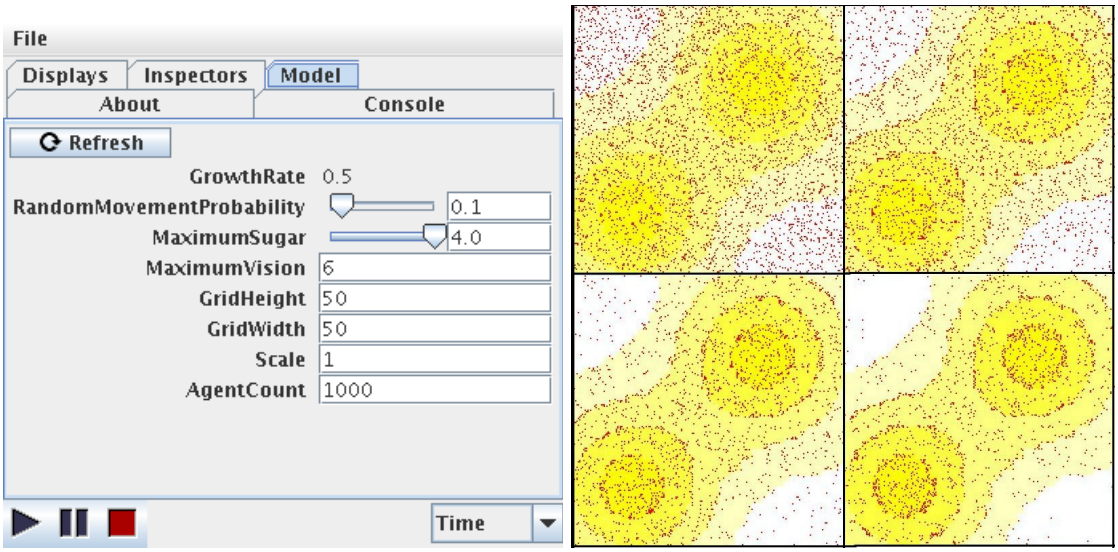

Figure 2: Functionality and engineering testings.

2.2 Engineering

The engineering testing consists of both tests on the correct implementation of the steppable functions, and tests of whether the expected output is the same as the theoretical output. The new rule implemented is Distribution3D1 meant to gather the data for the wealth distribution of the agents. The rule has been incorporated into the schedule of Sugarscape after the growth rule (RuleG), to test whether or not it collects data in the correct manner (it had trouble with obtaining an agent's information from the Sparcegrid3D agentgrid, and is not collecting data correctly) 0 - 10: 10 - 20: 20 - 30: 30 - 40: 40 - 50: The output pattern of the agents do correspond with the expected patterns around the latices of the different sugar levels (Figure 2b). The addition of the distribution rule also tests for the adaptability of the program to encompass new rules in its runtime schedule. Further rules, such as reproduction and cultural exchange should also be viable, given the problem with the accessibility of the agents' private fields.

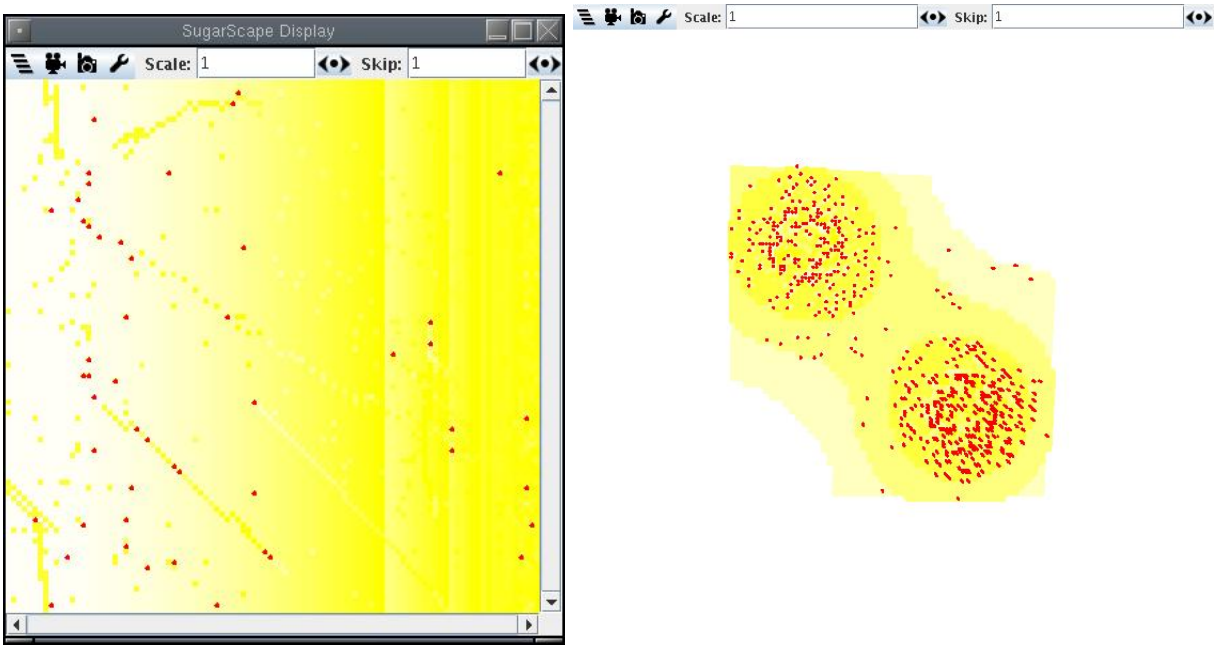

Figure 3: a and b.

3 Results

3.1 Random Initial Positions

In the instance when each agent begins the first step of the program with random positions, the agents demonstrate a tendency to gather at lattices under the finite growth rule (RuleG1). Initially the agent migration patterns showed some error (Figure 3a), such as a tendency to move back and forth in the horizontal direction, or a tendency to bypass closer cells with equal amounts of sugar. The pattern of moving to lattices was also only visible when the world was scaled to the size of 200 X 200 and up to 4000 agents. However, with the implementation of a distance check in the agent's step() method, which determines the cell of the best value that is closest to the agent, the agent migration correlated with expected patterns. The agents start with randomly assigned positions on the grid in the form of (x,y). With each step the agent will pick the optimal position and continue to advance until the pattern in Figure 3 occurs. This is the case when the growth rule is infinite. When the sugar growthRate is a limited number, the agents move to the poles, where the greatest amount

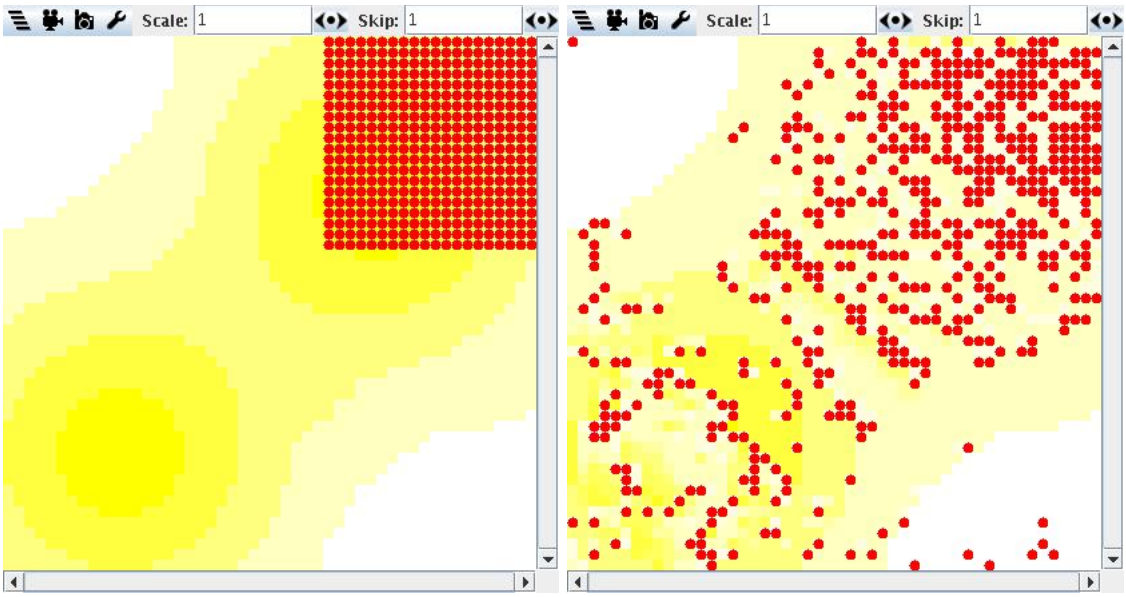

Figure 4: a and b shows agent migration in wave pattern.

of sugar is available (Figure 3b).

3.2 Initial Pattern

The initial positions of the agents have been changed to a square within the grid of the sugar scape world. The size of the square's side length is roughly 40 percent that of the grid, starting in one of the corners. The migration pattern should show waves of agents (Figure 4a-b) moving from one pole towards the other pole. The sugar amount will lower with each step of the agent, causing the agents to move forward (towards the other pole), because the amount of sugar in front of them will be greater than the amount behind. The agents show the pattern of movement, but since the toroidal grid is implemented, some agents wraps around to the other pole through the corner rather than forward. The next step is to make the grid non-toroidal.

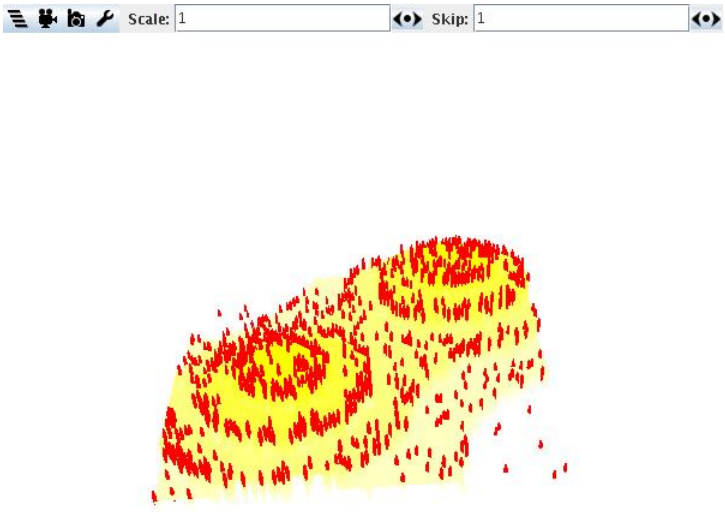

Figure 5: 3D representation shows same migration pattern.

3.3 3D Representation

The 3D representation draws the agents as ellipses instead of spheres. The representation is the in the z direction, with different levels of sugar the correspond to different z values. The result is a graphics representation consistent of different latitudes, which symbolize the changes in density of sugar in the area. The poles are the highest on the map, and the area of no sugar are equal to ground level. As agents consume the sugar at a position, the level of the position becomes zero (Figure 5), and the contrast shows which positions have been harvested by the agents.

4 Conclusion

The basic statistics of agent movement, wealth distribution based on the accumulation of resource were recorded and displayed through the use of screen shots and graphs. The array

of events and changes in the agent world were displayed through a Java applet, updating itself with each iteration.

The patterns of migration in the agents show interesting insight into progression of a population when a limited resource is in question. When each member of a society acts for its own benefits, an aggregate pattern of either gathering at lattices or wave movement is distinct. Through decentralized rules, the society creates seemingly centralized activity.

References

[1] Axelrod, Robert, The Complexity of Cooperation, Princeton University Press, Princeton, New Jersey, 1997.

[2] Axtell, Robert and Epstein, Joshua M., Growing Artificial Societies, Brookings Institution Press, Washington, D.C, 1997.

[3] Introducing Sugarscape, The Brookings Institute, 20 Jan 2006. ¡http://www.brook.edu/es/dynamics/sugarscape/default.htm¿.

[4] Simon, Herbert A., The Sciences of The Artificial Third Edition, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1996.

[5] Taber, Charles S. and Timpone, Richard J., Computational Modeling, Sage Publications, Thousand Oaks, 1996.